# NBA on TNT NLP: Natural Language Generation of Textual Commentary for NBA Basketball Games

**Jacob Bustamante** and **Nathaniel Chamness**
Computer Science
California Polytechnic State University
San Luis Obispo, CA 93407
{jbusta02,njchamne}@calpoly.edu

*Abstract*— **Basketball play by play text information online is typically dry stats indicating what occured in a play. We aim to add interesting facts and color commentary to improve the experience for fans following online. Our system automatically generates this for an NBA basketball game. The generator uses a bidding structure to select the most relevent and noteworthy comments for each game event, then chooses from a library of predefined sentence templates associated with that bid event. We validate our output through a short survey, the results showing that the average of scores given to our commentary are higher than the commentary of other major sports sites, an improvement in the status quo.**

## I. INTRODUCTION

We came up with the idea for NBA commentary generation as a part of a course on Natural Language Processing we both were taking at Cal Poly. Because we were interested in the statistics of sports we wanted to combine that with the topic of text/story generation so that we could use it as an NLP project. Currently the information displayed in the play by play section of an NBA game on a major website such as ESPN, Yahoo, or Fox Sports is typically very bland, so we wanted to see if we could come up with colorful short narratives that mimicked what an announcer might say if reporting on TV or the radio. We knew that there would be limitations to our system as we would not have knowledge of passing, defensive match-ups, and the difference between a posterizing dunk vs a relatively tame dunk as our data source would categorize them the same. Besides this, we were convinced that we could still come up with creative statistics using knowledge of past plays and the descriptions we were given.

## II. PROBLEM

As mentioned, much of the live play by play commentary on sports websites is very bland and robotic. It's meant to be presented this way, but we thought that we could offer an improvement by presenting the reader with a little narrative generation. Examples of currently plays might read, "LeBron James missed 3-pt. Jump Shot" and "Defensive Rebound by Chris Bosh" give the reader enough description to understand what happened on a specific play, but they don't infer any knowledge from previous plays or games. We wanted to be able to say "Lebron James missed 3-pt. Jump Shot, he has only been shooting 20% from beyond the arc the past 3 games." Using our knowledge of all the plays and games in a season to come up with some relevant statistics to display. We came up with this idea from watching NBA broadcasts ourselves, this is a kind of statistic a commentator will mention during the game.

## III. RELATED WORKS

In this section, we review research that inspired us relevant to our problem of automatic commentary selection.

### *Automated Story Selection[9]*

Lee, Bulitko, and Ludvig created a system called the Sports Commentary Recommendation System (SCoReS) that will automatically suggest stories for commentators during games.

Their system uses a ranker that is given a game state and will return a ranked list of stories based on how appropriate they are for that game state. Their system is quite different from our approach because they are not actually generating commentary, but instead suggesting an existing story from a database to tell. Though their approach is not exactly the problem we are looking to solve, we still liked the idea of their ranking based on game state and figured we could use the idea and adapt it for our system. Essentially we would substitute in our commentary generation module where they would've returned an existing story.

### *Enhancing Real-Time Sports Commentary[11]*

This paper provided a formal analysis of sports commentary to explicitly model the dramatic aspects to enhance the emotive qualities of simulated sports commentary. The most important bit of their paper to us was an example flow diagram of the architecture, how they proposed going from an action occurring to the natural language generation. It flows as follows:

1) Action Event Stream (Event Primitives): An action event occurs
2) Concept Event Stream (Assertions): Consequences of the action
3) Ontology Update (Update Primitive and Asserted Fact Base): Update the database to reflect the action that just occurred

4) Theme Management (Rules): Make some insights based on the action
5) Content Selection (Rules): Action, concept and theme-based commentary nuclei are select for expression according to rules and active themes
6) Utterance Generation (Natural Language Generation): Creating the narrative output for the viewer

We borrow from their general flow diagram to dictate how we go from raw events in our database to our generated narrative.

### *SCIgen [10]*

SCIgen is an automatic CS paper generator that is infamous for getting nonsense papers it generated accepted into numerous conferences. The sentences produced by the generator are gramatically correct and are made to look relevant to computer science. The generator used templates to generate its sentences. One template, for example, is:

*SCI_AB_A_START in fact, few SCI_PEOPLE would disagree with*

The ”SCI_PEOPLE” flag in the template gets replaced by some chosen group of people related to science. In one of the generated SCIgen papers, this template was used to generate:

*In fact, few hackers worldwide would disagree with*

Here, the word ”hackers” was used to replace the ”SCI_PEOPLE” flag. This template structure inspired the use of templates for our own commentary generator, using possible commentary strings as templates and game data as the variables to fill in the templates with.

## IV. Retrieving the Source Data

We spent a bit of time searching for sports website pages that could be scraped for past and possibly live game data. Originally we planned on using a combination of gamepages from Fox Sports[7], Yahoo Sports[6], and CBS[8] Sports to fill the gaps in the information that each provided. Upon further inspection though, the Fox Sports page source included a raw JSON string of all the datapoints we could've wanted. Table 1 shows an example of a play in that JSON string, hundreds of these would make up a full game. From that format it was trivial to insert most of the play information into a SQLite database file.

After parsing the JSON, we needed a persistent data store to store the information for each play, players, games, etc.

### *A. Motivations for SQLite*

We had multiple reasons for choosing SQLite as our data store for our project. We needed a persistent data store so our first thought was to create a Python class to store it all and then pickle that into a file so that it would persist. This would've involved figuring out a way to query the data that we gathered to make meaningful stats. Because we both knew SQL and knew that it would be easy to translate our thoughts for statistics into SQL queries we decided to look at MySQL and SQLite. Ultimately we chose SQLite because it lived in a single file and provided the necessities we were looking for. It was easy to setup and didn't require a MySQL server to be running on our local machines.

TABLE I
An Example of a play in the FoxSports JSON string

| Field | Value |
|---|---|
| quarter | 1 |
| time-minutes | 11 |
| time-seconds | 1 |
| id | 7 |
| first-name-1 | Kawhi |
| last-name-1 | Leonard |
| global-player- | 512591 |
| player-id-1 | 4896 |
| global-team-code-1 | 24 |
| team-code-1 | 24 |
| global-player-id-2 | |
| player-id-2 | |
| points-type | 2 |
| event-id | 4 |
| event-description | Field Goal Missed |
| detail-id | 66 |
| detail-description | Bank Shot |
| blocked” | false |
| x-shot-coord | 8.9 |
| y-shot-coord | 2.3 |
| second-chance | false |
| off-turnover | false |
| visitor-score | 4 |
| home-score | 0 |
| textual-description | Kawhi Leonard misses a bank shot from 9 feet out. |

## V. Commentary Generator

The generator was implemented as a collection of python files that get run for a single given NBA game. The program then generates textual commentary at runtime and outputs the full text to the user. Our description of the generator will be broken down into four main categories: The play-by-play event iterator, the bidding structure, the color commentary, and the sentence generation templates.

### *A. Play-by-play Event Iterator*

The scraped play-by-play commentary for a given game consists of short sentences such as ”Lakers with an offensive rebound” and ”Kobe Bryant makes a driving layup from 1-foot out.” These comments are parsed for all relevent data and labelled with a specific general event type. These general event types include: shot, assist, block, foul, rebound, steal, substitution, timeout, jumpball, and free throw. Our commentary generator consists of a main loop, the play-by-play event iterator, which iterates through these labelled events and calls a specific comment generator function according to the given event label. For example, in figure 1, an event labelled as a shot was processed by the play-by-play event iterator. The event is then passed to the appropriate comment generator bidding functions, which will be explained in the next section.

The iterator also calls specific comment generator functions both at the beginning of the game and at the end of the game. These are the gameStart and gameEnd functions, which generate game preview and summary text commentary.

## B. Bidding Structure

The bidding structure is where most of the complex logic takes place. Each event is associated with a collection of bidding functions. A single bidding function takes in the current event data and returns a bid score between zero and one and a generated comment string. The bid score rates how relevent and notable the current event is to that bidding function. The generated comment string is a line of commentary related to the specific bidding function and data from the current event.

For example, the current event being processed might be a 'fastbreak made basket'. A fastbreak occurs when a team wins possession of the ball then quickly gets to the opposite court to make a basket. This event will trigger the shot event. Two existing bidding functions associated with a shot event are the madeBasket bid function and the fastbreak bid function. When the event triggers those two bidding functions to be called, madeBasket will return a bid score 0.3 and a string such as, "Kobe Bryant makes the basket." The bid score is relatively low because made baskets are common occurences in a game. The fastbreak function, however, will return a bid score of at least 0.7 and a string such as, "After the rebound by Steve Nash, Kobe Bryant scores a fastbreak basket." The bid score is relatively high because a fastbreak is a more notable occurance in a game. The program will pick the fastbreak function because of the higher bid score, then the functions comment string will be output as the generated commentary for the given event.

## C. Color Commentary

Each event is structured to work with bidding functions in the same way as the given example above for the shot event. Apart from those bidding functions, all events are also associated with color commentary bidding functions, which we called meta-bidding functions. These meta-bidding functions are used to generate color commentary strings such as, "Amazing! That 80-foot shot by Trevor Ariza was the longest shot made all season!" and "These two have been playing great together for 50 games this season. They've linked up for an assist an average 2.0 times per game." Similar to the normal bidding functions, the program will pick the meta-bidding function's color comment string with the highest score. That color comment string is then output directly after the normal bidding function's comment string. If all meta-bidding functions return a score of zero, however, then no color comment string is output.

Many of the meta-bidding functions check that the current play matches a defined scenario, such as, shooting the first shot of the game, or a player fouling too many times getting into foul trouble. These only require running basic logic with the current game data. Some of the more complicated scenarios, however, require stats about past games of the current season. We implement this by running queries on our SQL database, which holds game data for all of our scraped games.

## D. Sentence Generation Templates

Each generated comment string is formed using sentence generation templates. These templates are strings with variables in them to be replaced by words specific to the event that the sentence is being generated for. Every bidding function and meta-bidding function is associated with at least one set of these templates. For example, one of the sentence templates for a made basket is this python tuple:

*[("%s's %s shot is good.", ['primary_player', 'shot_type'])*

The "%s" flags get replaced by the current events "primary_player" value and "shot_type" value, accordingly. A generated sentence using this template would be:

*"Kobe Bryant's jump shot is good."*

The bidding function chooses one sentence template from the set of templates associated with it to generate a commentary string. Each template sentence in a set is written to convey the same exact event, but phrased differently. The bidding function chooses a template from this set at random. This is done to increase the variety of phrases that the program will generate. A made shot, for instance, will not always produce "Kobe Bryant's jump shot is good.", but can also produce "Kobe Bryant makes the basket."

## SQL Queries

SQL really made it easy to add new stats to our commentary. Our process started with us coming up with a scenario that we would want to detect during a game to provide commentary for. One of our first notable scenarios was if a player made an assist we wanted to check if the two players involved averaged two or more assist plays per game. Then we would add a little fact about that to our commentary. We came up with two assists as our cutoff after running some queries to discover that there weren't many pairs who reached this amount with at least five games played together. SQL made it easy to then run the query to find the average assists per game between that player pair. That query is listed in Listing 1.

Listing 1. Sample SQL Query for avg assists between player pair

```
SELECT p_player, s_player, avg(assists)
FROM (
  SELECT p_player, s_player, game_id, count(*) assists
  FROM play_by_play
  WHERE play_type = 'assist'
   AND secondary_player = play['secondary_player']
   AND primary_player = play['primary_player']
   AND id < play['pbp_id']
  GROUP BY game_id, secondary_player, primary_player
)
GROUP BY primary_player, secondary_player
ORDER BY avg_assists DESC
```

*Game Preview & Game Summary*

Both a game preview commentary and a game summary commentary are generated by slightly modified bidding functions that are run at the beginning and end of the play-by-play event iterator accordingly. The game preview bidding function, rather than pick the one highest bidder, builds a group of sentences to form the multi-sentence game preview commentary. The function begins with a simple comment on which teams are playing and which of them is the home team. The function then adds a comment on both of the team's win-loss record. This is followed, finally, by the starting lineup for both teams. The game summary bidding function adds an ending comment on the final score and which team won the game.

## VI. Evaluation

The goal of our commentary generator was to provide a play-by-play commentary that is more interesting than the existing play-by-play commentary on websites such as, ESPN, Yahoo, and CBS Sports. In order to perform a proper evaluation of this goal, we conducted user testing. We set up our test to include 3 play by play commentaries, one of which we generated, one from CBS, and one from Yahoo. We did not tell our subjects which was which to try and minimize bias. We also didn't tell them in the email we sent out that we were generating one of the commentaries, we just said that we were doing a survey on NBA commentary and wanted to get their opinion on a few of them.

Our intention of this evaluation was to get some quick feedback on whether or not the direction we were taking was a good one and whether our generated commentary offered any improvements over some of the major sports websites. Our results were promising.

For this initial evaluation we received the opinions from 6 people who gave their ratings on easiness to follow, level of interest, descriptiveness, and how enjoyable each of the commentaries were. Given abitrary scores of 1-5, our commentary averaged just over a 4 on the average of every category per person. The two other play by plays scored 3.48 for CBS and 3.14 for Yahoo respectively so we consider our score a noteworthy improvement. Where our commentary really shined was with its descriptiveness, our evaluators gave it a score of 4.58 out of 5, compared with a 3.67 and 3.5 for the other two sources.

We're proud of these results and believe that our metrics can be improved if we implemented some of our other ideas. A few of our ideas are outlined in the next section.

## VII. Future Work

While working, we had a lot of ideas of improvements and add-ons to our system that would've improved the quality and density of our commentary. A few examples are:

- More stats based on the performance of a team. We currently are aware of when a team goes on a winning/losing streak, but knowledge of their playoff hopes and how well they are playing against teams with winning records would be great to include during down time between plays.
- Awareness of player quality would make the commentary seem much more personal. We could output that LeBron James is an invaluable asset to his team during a time in the game where he is playing well. We would know this by looking at the stats he regularly puts up and seeing that it is above average.
- Mention of players who are injured and how a long term injury has affected a team. Russell Westbrook and Kevin Durant's absences for much of this November completely changed the OKC team and were often talked about in the commentary. There are data sources online to get this information.
- Our system currently generates basic game preview and game summary commentary. We would like to improve upon this to create more in-depth and meaningful comments. A game summary, for example, would occur both at half-time and at the end of the game. The improved summary would include the most notable events of the game, any deviance from expected team performance, and interesting game statistics.

*Expandability*

It is worth noting about the exandable nature of the structure of our commentary generator. Both the bidding structure and comment generation templates are built with the ability to be easily expanded upon. A new bidding function that could produce a new type of comment can easily be developed and placed amongst the existing bidding functions for a certain event. The system calls from that list of bidding functions and returns the highest bidder's string. Similarly, a set of comment generation templates is simply a list of Python tuples, made up of strings and the values to fill variable flags within those strings. New templates can be continuously added in order to increase the vocabulary of the generator.

## VIII. Conclusions

Our commentary generator was shown to successfully produce interesting commentary for an NBA basketball game. Its general structure is made up of a SQLite database full of scraped NBA data, a bidding comment structure associated with labelled game events, and a set of comment generation templates to build the textual commentary output. The output includes color commentary along with the basic play-by-play commentary in order to provide a more interesting and useful commentary to readers, compared to the current play-by-play found on other websites. There is definitely more work that can be done to improve upon the generated commentary, but the current system has already proved to generate commentary that is more interesting and enjoyable than existing systems, according to our user testing evaluations.

## IX. Example Output

The following are exerpts from the commentary generation of the December 11, 2013 NBA basketball game between the New York Knicks and the Toronto Raptors.

```
Q1 12:00 Knicks 0-0 Raptors
The Raptors will be playing the Knicks
here in New York.
The Knicks are currently with a 36-45
record, while the Raptors stand with a
48-33 record.
The starting lineup for the New York
Knicks is Amar'e Stoudemire, J.R. Smith,
Cole Aldrich, Iman Shumpert, Pablo
Prigioni.
The starters for the Toronto Raptors are
Kyle Lowry, Amir Johnson, DeMar DeRozan,
Terrence Ross, Jonas Valanciunas.

Q1 12:00 Knicks 0-0 Raptors
And here's the tipoff to start the
game. Pablo Prigioni on the Knicks gains
possesion.

Q1 11:37 Knicks 0-0 Raptors
A missed shot from Cole Aldrich.
And there's the first shot attempt of
the game.

Q1 11:35 Knicks 0-0 Raptors
Amir Johnson rebounds the ball.

Q1 0:06 Knicks 21-27 Raptors
Kyle Lowry puts two points on the board
for the Raptors with a driving jump
shot. Patrick Patterson with the assist.

Q3 1:35 Knicks 69-70 Raptors
There's a foul called on Kyle Lowry.
Kyle Lowry's fourth foul puts him in
foul trouble with still over a quarter
remaining.

Q4 8:53 Knicks 78-77 Raptors
Steve Novak comes off the bench for for
Patrick Patterson.
Scoring 32 of the team's 77, the
Raptors bench are having a great showing
tonight.

Q4 0:00 Knicks 95-92 Raptors
Game over. The Knicks end the game with
a 95 to 92 victory over the Raptors.
That ends the night here in New York.
```

## References

[1] "Beautiful soup," http://www.crummy.com/software/BeautifulSoup/bs4/doc/, accessed: 2014-11-01.

[2] "json - json encoder and decoder - python 3.4.2 documentation," https://docs.python.org/2/library/json.html, accessed: 2014-11-01.

[3] "Nba schedule - insidehoops.com," http://www.insidehoops.com/schedule.shtml, accessed: 2014-11-01.

[4] "Spurs vs. heat — the finals — 2014 nba playoffs: News, highlights, and video," http://www.nba.com/playoffs/2014/finals/, accessed: 2014-11-01.

[5] "Toronto raptors vs. new york knicks - live nba gametrax - fox sports - april 16, 2014," http://www.foxsports.com/nba/gameTrax?gameId=2014041618, accessed: 2014-11-01.

[6] "Miami at san antonio," 2014, example of Yahoo Sports NBA play by play data webpage. [Online]. Available: sports.yahoo.com/nba/miami-heat-san-antonio-spurs-2014061524/

[7] "Miami heat vs. san antonio spurs," 2014, example of FoxSports NBA play by play data webpage. [Online]. Available: foxsports.com/nba/gameTrax?gameId=2014061524

[8] "Nba basketball play by play," 2014, example of CBS Sports NBA play by play data webpage. [Online]. Available: cbssports.com/nba/gametracker/playbyplay/NBA_20140615_MIA@SA

[9] E. A. L. Greg Lee, Vadim Bulitko, "Automated story selection for color commentary in sports," in *TCIAIG*, 2013.

[10] D. A. Jeremy Stribling, Max Krohn, "Scigen - an automatic cs paper generator," 2005, mIT CSAIL - PDOS Research Group.

[11] T. C. Martin Rhodes, Simon Coupland, "Enhancing real-time sports commentary generation with dramatic narrative devices," in *ICIDS*, 2010, pp. 111–116.

## APPENDIX

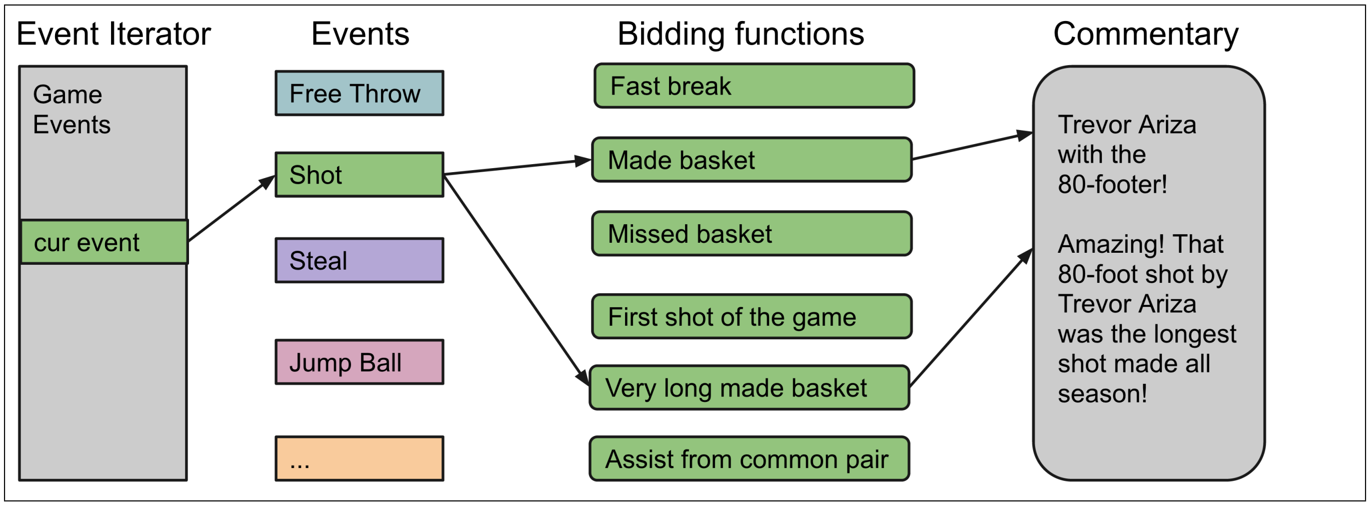

Fig. 1. Diagram of the process flow of a shot event through the commentary generator. The shot event triggers all the bidding functions associated with it the be called. Both the made basket function and the ver long made basket function are chosen and generate the resulting commentary text on the right.